Predicting Post-Release Software Security Fault Discovery Using Expert Judgment Elicitation and Bayesian Computed Effort-Dependent Models

Reuben A. Johnston
The George Washington University
reuben@reubenjohnston.com

ABSTRACT

Publicly disclosed, post-release mobile device software security faults experienced growth from almost zero to over 600 in the years 2008-2012, negatively impacting the reputation of software vendors and reducing customer confidence in the quality of software security provided. To mitigate this problem of post-release user discovery of security faults, software company decision makers have at minimum the following investment alternatives: (1) *pre-release process quality improvement*, (2) *modifications to release rate strategies*, (3) *increased resources for rapidly addressing post-release faults*, and (4) *methods (or features) which would increase the analysis effort (i.e., the energy expended in software security assessments) required for discovery*. This paper proposes two phases of research, the result of which will hopefully provide new, or improved, techniques for evaluating security investment options software company decision makers have available. Phase I will explore the utility of Bayesian computed generalized Non-Homogeneous Poisson Process (NHPP) models on post-release user discovery of a subset of software security faults (i.e., those belonging to what the authors consider to be the higher risk categories). The intent of this phase will be to model performance evaluation on this subset of fault types and also to determine whether Bayesian computation benefits the data sparseness and sequential release problems. Phase II will evaluate the effectiveness of various security investments (in design, development, verification, and operation periods) on the analysis effort required for public discovery of software security faults. As there is no public analysis effort information available, phase II will utilize expert judgment elicitation techniques, via the Cooke classical model, in gathering the predicted times to discovery events and their associated analysis effort for various software release scenarios. Bayesian computed analysis effort-dependent NHPP models will be introduced, and their performance will be evaluated against the generalized NHPP models from Phase I. The target audience for this research includes: software companies, academia interested in security applications of software reliability techniques, and organizations evaluating decision alternatives for high security software.

Categories and Subject Descriptors

D.2.8 [**Software Engineering**]: Metrics – *process metrics, product metrics*; G.3 [**Probability and Statistics**]: Reliability and life testing, Stochastic processes.

General Terms

Management, Measurement, Reliability, Security.

Keywords

software security, security metrics, software security vulnerability, software testing effort, software security analysis effort, Vulnerability Discovery Model (VDM), Software Reliability Growth Model (SRGM), Non-Homogeneous Poisson Process (NHPP) model, effort-dependent model, expert judgment elicitation, Bayesian analysis.

1. INTRODUCTION

Publicly disclosed, post-release mobile device software vulnerabilities[1] experienced growth from almost zero to over 600 in the years 2008-2012 (Figure 1), negatively impacting the reputation of software vendors and reducing customer confidence in the quality of software security provided. With the elevated quantities of mobile devices sold in this time period, more than 434,000,000 in 2011 alone [5, 34], the occurrence of any serious software security fault poses significant risk of financial loss. Consider the following relationships as one example of the vendor, customer financial dependencies: (1) externalities, such as customer losses due to the exploitation of these software security faults (e.g. losses from identity theft and fraud), are negatively correlated with vendor reputation, and (2) vendor standing with the customer and future revenues exhibit a positive correlation. These relationships illustrate the business need for maintaining security quality reputation with customers [37]. Software vendor decision makers must balance product lifecycle security investments, as mitigation to these external customer security risks.

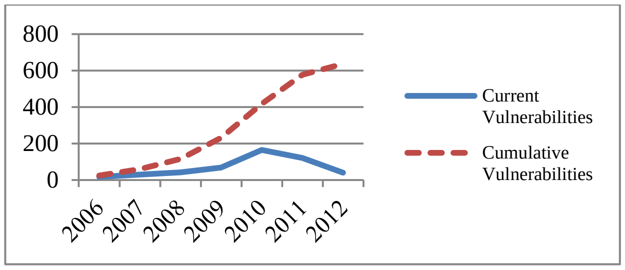

Figure 1-Mobile Software Vulnerability Trends [15]

Software company decision makers have several investment alternative areas available for addressing the need to maintain security quality reputation with customers. They include: (1) *pre-release process quality improvement*, (2) *modifications to release rate strategies*, (3) *increased resources for rapidly addressing post-release faults*, and (4) *methods (or features) which would increase the analysis effort required for post-release fault discovery*. Empirically based prediction and risk estimation methods are necessary for decision makers to allocate security

[1] A vulnerability is, "an instance of a mistake in the specification, development, or configuration of software such that its execution can violate the explicit or implicit security policy" [20, 30].

investment resources effectively between these and other potential areas. Two categories of modeling techniques have emerged for prediction and risk estimation in the area of process quality improvement, Vulnerability Discovery Modeling (VDM) [2] and vulnerability prediction modeling. Optimal release rate strategies for security have been investigated [26]. The authors are not aware of any literature introducing methods for decision makers to evaluate the effectiveness of security investments in the third or fourth areas listed above, although studies have evaluated the effects of rapidly addressing post-release faults (to control user perceived quality) in the general fault context [24], and one study attempted gathering analysis effort only [36]. This research proposal introduces techniques for evaluating investment alternatives which influence post-release analysis effort. It will also explore the utility of certain models and techniques in the two pre-release quality improvement investment decision making categories mentioned above. These two categories are better introduced in the following two paragraphs.

Vulnerability Discovery Modeling uses various software reliability modeling techniques on post-release security fault discovery events (i.e., the point in time when a security fault becomes known) [30]. It is utilized in the prediction of post-release security fault discovery events for future software releases, enabling decision makers to better allocate quality improvement resources as security risk mitigation [2]. These models provide insight into the quantity and timing of vulnerabilities to be discovered following the next software release, based on the assumption that there are no changes to the existing process (i.e., the process used over the period of time covering when the historical data, used as input to the model, was generated). Decision makers can use this to decide whether investment deviation in either process or resources is necessary. VDMs emerged from traditional software reliability techniques and appeared in academic journals around 2002 [4]. They were researched significantly in the literature in the years 2004-2008.

Vulnerability prediction modeling is based upon software characteristic information gathered during the design, development, and pre-release verification phases [10, 35]. The general goal of this type of modeling is to use software characteristic information as a basis for determining indicators to vulnerable software components within a system. It is highly desirable to prevent the release of security faults (i.e., security quality improvement). Pre-release security investments, directed towards verification of the higher risk design elements[2], are intended to reduce post-release user fault discovery [10]. Given the large size of modern Operating System (OS) software projects, these techniques allow for efficiency in software verification, which is an important consideration to meeting schedule and cost requirements. They emerged from traditional software fault prediction models, and were proposed in the literature around 2005 [12]. They have been researched increasingly in the years 2007-present.

There are several areas in the current software security literature which remain to be addressed or could use improvement. These include: (1) VDM techniques which do not violate SRGM assumptions (i.e., do not combine fault data from multiple releases) [25, 30]. (2) Evaluation of the applicability of certain methods on vulnerability category subsets, as it is believed that vulnerability categories should not all be considered of equal importance. (3) A need for exploration of additional data gathering techniques for inputs to the models, as the data is sparse and the sources are inconsistent [25, 30]. (4) Proposing a means of normalizing models with respect to post-release analysis effort (as post-release analysis effort can be significantly different in each of the many vulnerability categories, between software releases, and amongst the various product types [30]). (5) Quantitative approaches for evaluating security investment alternatives.

Towards reducing some of these literature gaps, this research is being proposed as two phases. The first phase will be an observational study with the goal to evaluate the effectiveness of using generalized NHPP as a VDM. Data for this phase will be gathered from mobile software major versions released (this will at least include firmware released for a particular group of devices running the Android operating system) within the years 2008-2012 and will only include a subset of the vulnerability categories (those which enable privileged[3] execution of user loaded code). The data within this timeframe should prove to be a useful disruptive technology example for study, as it was a highly competitive new market, with compressed release schedules to deliver products. The research is intended to explore solutions addressing research gaps 1 and 2 above, and will help to develop computation techniques which will also be used in the second phase.

The second phase will be an experimental study, using expert judgment data elicitation from software security researchers, to evaluate the effectiveness of several security investment alternatives in various software release scenarios. It is intended to explore solutions to research gaps 1-5. In addressing 4, it will introduce and demonstrate the application of software analysis effort-dependent NHPP models, as vulnerability discovery models. Expert judgment elicitation is necessary for the data gathering, as there is no public information available for security analysis effort (these techniques are also to be evaluated as a solution alternative for gap 3). Ultimately, it is hoped that the phase II research will provide computational techniques and justification for future empirical research which would gather analysis effort data from actual projects, as well as provide influence to vulnerability database maintainers to capture estimated analysis effort information with future vulnerability entries.

The stakeholders for this research include: software companies (primarily those developing Operating Systems and original equipment manufacturers), academia interested in security applications of software reliability techniques, and organizations evaluating software decision alternatives for high security uses. Software companies will be most interested in the results and recommendations from this research because, as already mentioned, their decision makers need to invest appropriately in security quality. This research will develop, demonstrate, and evaluate methods useful for estimating risk to company reputation during design alternative decision making, enabling managers to make informed security investment decisions on security investment options in the design, development, verification, and maintenance phases of the software product lifecycle.

The remainder of this paper is organized as follows. First, there is a brief section on the author's solicitation for research proposal

[2] Pre-release security analysis priority should be granted to software components with the highest likelihood of error [10].

[3] Privileged code executes with higher operating system security level permissions [12].

advice. This section is followed with a literature review on the prior work relevant to this paper. Next are sections in which the research questions, hypotheses, and objectives are listed and discussed. Research approach is then covered and this is followed by brief sections discussing metrics and also the data analysis. Lastly, there are sections mentioning some validity threats, with mitigation proposals, and a status summary.

2. SOLICITATION FOR ADVICE

This section contains a prioritized list of three questions for advice solicitation (with l being the highest priority). The questions are: (1) Numerous phase II variables (Table 1) will make the scenario combinations high in number. Are there certain subsets of variables which can be eliminated to reduce scope? Are there additional suggestions for metrics? (2*)* Very low fault counts are anticipated for vulnerabilities in the categories of interest (possibly 0-3 per release over three consecutive annual releases). How can research validity be ensured? (3) What is critical to acceptance of data elicited via expert judgment? In the next section, a brief literature review of relevant research is discussed.

3. RELEVANT PRIOR WORK

The general goal of software security modeling is to utilize empirical data to predict security fault information useful to industry decision makers. One category of these, termed Vulnerability Discovery Modeling (VDM), uses various software reliability techniques on post-release security fault discovery events. The Alhazmi-Malaiya Logistic model (which applies a slight variation to the Verhulst logistic growth model) [1-3], and the Musa Okumoto Logarithmic Poisson NHPP have the most popularity in the literature [2, 3, 25, 28]. Additional notable models include the Weibull distribution based model [17], and the Goel-Okumoto model [32].

There are several critical problems with VDM and their use in modeling software security fault discovery, the first of which is data quality and the sparseness of data [25, 30]. Others include the need to normalize detection events for effort and the issue that post-release operational environments can be drastically different between product types [30]. The improper assumption for static code across releases was pointed out as being a deficiency in many VDM studies [25, 30].

Vulnerability databases (including public and private sources) suffer from "chronological inconsistency", "inclusion"[4], and "documentation differences" [29]. In addition to these problems is the issue of missing data (i.e., from the public vulnerability databases), due to the effects of companies withholding information on privately disclosed vulnerabilities. There are also the hostile tendencies of some companies threatening legal consequences to public disclosures (post-release analysis effort may have dependencies with this)[5]. Due to this, public databases may very well be missing known discovery events. Bayesian analysis is a technique for addressing the issue of data sparseness.

Post-release operational environments can be significantly different for the various software product categories, as compared to the pre-release testing environments. As such, there is a need for normalizing post-release detection event times to software security fault analysis effort [29]. The differences in analysis environments is more than likely related to the return on investment for detecting vulnerabilities from each of the various categories. Effort based models, or models including effort as an input are a potential solution to this problem of data normalization.

The violated assumption of static code in the improper use of SRGMs, across multiple code bases, is another issue in the literature. Some of the VDM research has combined events across multiple releases, to address data sparseness. Software within certain studies has been confirmed to change significantly between successive releases [25]. One VDM solution has been proposed for the sequential release problem [19]. There is still a need in the literature to expand upon this initial research.

Stochastic processes are natural modeling techniques for phenomenon whose rate of occurrence of events changes over time [33]. One such technique, the generalized NHPP model, was proposed for use in security modeling with three basic assumptions: (1) that a software product has a finite quantity of security faults introduced per release, (2) the time to security fault discovery is stochastically distributed, and (3) all discovery times are independent random variables. The authors suggest applying different statistical distributions (e.g., hyper-Erlang, exponential, logistic) for the rate of occurrence of events, to model various scenarios for how software security fault discovery changes over time [13, 26, 27]. This is very similar to another study using generalized NHPP's to estimate cyber-attack risk (cyber-attacks utilize software written to exploit software security faults) [33].

NHPP testing effort-dependent models were proposed for traditional fault prediction within the pre-release software verification phase [38]. These models are founded on a significant assumption that the rate of error detection is in proportion to the remaining (undiscovered) errors. As part of this, they also assume that the instantaneous software testing effort represents this proportionality [38]. The utility of these models has been explored extensively, most of which are referenced here [14]. Unfortunately for software security, post-release analysis effort data is not publicly available. To explore this, a study using the Cooke classical model for expert judgment elicitation on vulnerability discovery event analysis effort was performed [36]. Future work was suggested by the authors of the analysis effort study to repeat the process using software security analysts as the experts and this is expected to be a promising data source for evaluation of these models in the post-release security fault context. In the next section, the research questions, hypotheses, and objectives are introduced and discussed.

4. RESEARCH GOALS, OBJECTIVES

The following three questions are to be explored in this research:

Question 1a: Over time, is the rate of occurrence of post-release software security fault (i.e., vulnerability) discovery events non-homogeneous (e.g., increasing then decreasing)? Question 1b: Over consecutive releases, is the rate of occurrence of these events following the same pattern, albeit attenuated? The mobile vulnerability discovery rate illustrated in Figure 1 appears to not be homogenous, and is potentially decreasing over time (i.e., achieving reliability growth). Other studies also indicate decreasing vulnerability trends, for various software product categories over time [3, 29]. The use of the Musa-Okumoto Logarithmic Poisson NHPP model as a VDM has been explored

[4] Examples of inclusion would be determining which release a discovery event should be applied to for a security fault located within a code module evolved over sequential releases, or which modules should be considered part of the overall system under study (e.g., operating system including various external project libraries) [29].

[5] See the Cisco example discussed by Bambauer and Day [6].

in previous research and generalized NHPP modeling techniques have been suggested in a simulation study on software release frequency [26, 27].

Question 2: Are post-release software security fault discovery events influenced by security analysis effort? This relationship was proposed [23] and explored with the AMEL model [1]. They suggested that "Equivalent Effort" could be used as a proxy for analysis effort. It is defined as the summation of the total users of all systems times the percentage of the users using the system during that time. It has been challenged that there were unconfirmed assumptions used in the basis for the AMEL model's effort estimation [30].

Question 3: What security investment alternatives in the design, development, verification, and maintenance phases contribute most to post-release security analysis effort? It is suggested that information security investments for protecting technical design information (e.g., code obfuscation) will have a positive correlation with software reverse engineering effort [8]. Software reverse engineering is one of the two main tasks security researchers perform for post-release vulnerability analysis of software (i.e., they have to understand the software before they can locate security faults) [7]. It has also been suggested that reward strategies for post-release responsible disclosure[6] be studied [31].

The following five hypotheses will be evaluated in this research:

Hypothesis I: The rate of occurrence of post-release software security fault (i.e., vulnerability) discovery events is non-homogeneous and the rate is increasing then decreasing, or zero, following each major release. Over consecutive releases, the rate of occurrence of these events follows the same pattern, albeit attenuated. This is based on an assumption that adequate security investments are being contributed to the releases. With each successive release, there should be attenuation in the rate pattern [19], such as the one described above, and this is expected to continue until the rate becomes effectively zero defects per-release, or the software product is retired. Phases I and II will both explore questions 1a, 1b, and also test this hypothesis.

Hypothesis II: Vulnerability discovery event frequency is influenced by analysis effort and software security quality. These relationships (and question 2) will be explored in Phase II of this study, which proposes using security analyst's expert judgment elicitation to directly estimate the analysis effort for post-release vulnerability discovery events, as called for by [36]. This information, in addition to estimated time to discovery events, will be used as inputs to NHPP testing effort-dependent models [38, 39].

Hypothesis III: Technical information security process investments influence the analysis effort required for security fault detection events. It has been suggested that analysts searching for security faults many times require additional learning from external resources [7]. It is believed that creating a business process which limits access to technical documentation and has safeguards in place to prevent technical information leaks (e.g. CPU documentation relevant to security and other sensitive features such as low-level debugging control of the CPU), will provide increases in the post-release analysis effort required for vulnerability discovery. The phase II research will explore questions 3 and also test this hypothesis via expert elicitation.

Hypothesis IV: Information security features such as anti-tampering mechanisms, encryption, other features providing binary obfuscation (e.g., string mangling), and techniques which remove unnecessary information prior to release (e.g. debug features) will affect the analysis effort required for security fault detection events. The phase II research testing this hypothesis will explore question 3 and methods suggested in some studies (e.g., see [8]). Scenarios for providing various levels of binary obfuscation and unnecessary information removal will be used in the expert judgment elicitation.

Hypothesis V: Vendor provided features allowing for privileged user code execution (e.g., HTC's "developer unlock") will affect the levels of analysis effort expended by the open community[7]. This hypothesis is related to question 3 and will explore the utility of a feature provided by some mobile device vendors. It is an example of a maintenance phase provided feature option which may potentially reduce the open community's analysis effort (as there is less incentive to search for privilege escalation vulnerabilities when a feature providing this is available already).

The purpose of the Phase I study will be to explore hypothesis I by testing the theory through applying NHPP SRGM techniques, with an increasing then decreasing rate of occurrence (other patterns will also be evaluated), as a VDM. For products with adequate security investments, the trends should reflect attenuation in this pattern following the later releases. Computation techniques developed in this phase will be used in the second phase research. Lastly, characteristics gathered from the vulnerability trends in this study will have additional use as the basis for seed variable calibration questions in the Phase II expert judgment elicitation.

The purpose of the Phase II experimental research will be to explore the utility of the following security investment alternatives for decision makers: design information security process; design artifact obfuscation features; and features which would allow for user enabled (but vendor controlled) privilege escalation (hypotheses III-V). It is desired to understand the relationships between the alternatives above and the discovery events, with measures for corresponding detector security analysis effort expended (when examining software for software security faults) (hypothesis II). As analysis effort information is not available, this will be explored by an experiment utilizing expert judgment elicitation, from software security analysts practicing in the Washington D.C. metropolitan area. Prior distributions will be developed for parameters of detector analysis effort and post-release software vulnerability discovery event models. The desired outcome of this research effort includes: alternative VDM modeling techniques, demonstration of expert judgment elicitation for model input data gathering, demonstration of analysis effort-dependent VDM use for software industry decision maker's benefit, validation of some vulnerability prediction model techniques, recommendations for investment levels in software security to mitigate security risk, and hopefully, justification for future research utilizing actual project information. In the next section, the research approach is explained.

[6] Responsible disclosure means the vendor is notified prior to public disclosure of the vulnerability, allowing for patch deployment with disclosure (reduces user fault exposure) [30].

[7] In this paper, open community refers to non-commercial analysts (e.g., hobbyists) performing software security analysis.

5. RESEARCH APPROACH

As introduced previously, this research is being proposed in two phases. The first phase is an observational study on mobile device software security fault discovery events following release of a major version (i.e., incremental minor version, or patch, releases will not be considered). This research will illustrate Bayes inference for a Non-Homogeneous Poisson Process model, will explore questions 1a and 1b, and will test hypothesis I. It will use public software vulnerability discovery event and software release time data to develop the distributions.

The independent variable for this study will be security analysis time (following successive releases) and the dependent variable will be software security detection event times. For this study, software will be defined as all software executing on the device. This includes: Read-only memory (ROM) startup code within the micro-processor (i.e., ASIC ROM, or Application-Specific Integrated Circuit ROM, code), initial startup (i.e., boot-loader) firmware, operating system software, and application software. Any discovery of a software vulnerability allowing for privileged execution, if exploited, in ASIC ROM, boot-loader firmware, or the operating system levels will be considered a detection event. The scope of the study will at minimum include software on Android Operating System based mobile devices, in the time period 2008-2012 (note, the Linux kernel is considered part of this operating system). It is desired to include other mobile device operating systems such as Apple's iOS and Window's Mobile. However, data availability for these closed source products is inhibited and access will more than likely be difficult.

For data collection, the methodology will include gathering and consolidating all of the applicable vulnerability discovery events from public vulnerability databases (e.g., NVD, IBM X-Force). Additionally, it will require gathering of version control repository information for determining software version release dates. ASIC ROM and boot-loader firmware release dates will use the earliest model phone release date vulnerable to the discovered security fault. These dates will be necessary to calculate the detection time for each discovery.

The second phase is an experimental study which will focus on exploring the relationships between vendor security investment decisions (business decisions), released software characteristics (e.g., quality, size, complexity, design and development artifacts), post-release security analysis effort, and post-release security fault discovery event times (questions 2-3). This research will demonstrate the development of prior distributions for the parameters of software security fault discovery events and will use expert judgment results to develop the distributions. Expert judgment elicitation will utilize the Cooke classical model [9, 33] and will gather data from security experts with intimate knowledge of recent operating system software vulnerability history. These experts will be interviewed independently without knowledge of the other experts or their responses. Scenarios will contain multiple security investment levels defined for design of an operating system on a new consumer product experiencing exponential growth in sales during the first three years. Questions will be asked which obtain data such as information on vulnerability discovery (e.g., times or fault counts per some unit of time) and the associated security analysis effort following major annual releases. The results will be analyzed and are intended to be the basis for evaluation of hypotheses II-V.

Business decisions, analysis decisions, design characteristics, and external factors all have some levels of influence to the discovery events resulting from post-release security analysis of software. Figure 2 depicts the potential relationships among some of the categories of variables under study. In this figure, the control, confounding, and moderating variable categories fall to the left of the post-release security analysis rectangle and the dependent discovery events variable category is to the right. Table 1 contains the most relevant examples for each of the variable categories.

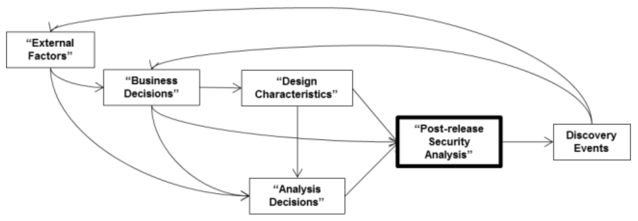

Figure 2-Potential influence of security business and analysis related decisions on analysis effort and vulnerability discovery

Multiple scenarios will be composed which demonstrate the various levels and combinations of the variables under study (several are discussed in detail within the metrics section). Some of the variables, for a particular group of scenarios, will remain static across the releases (e.g., pre-release analysis process or release frequency) and some may change in a predetermined fashion (e.g., code size growth per release, or units sold per release). The experts will be asked to specify 5th, 50th, and 95th percentile values for some of the calibration questions. The same is to be asked for the questions querying upon the dependent variables [33], such as the time to first and second discovery events and the associated analysis effort required, for each of the scenario combinations.

Some of the vulnerability prediction modeling methods in the literature are of interest to be included in the scenario developments for the Phase II experiment. For example, the use (and corresponding exclusion) of some of these techniques may be posed to the experts in the scenario alternatives, to explore their effects upon analysis effort and vulnerability discoveries. For these purposes, use of metrics such as privileged lines of code per module, and error prone construct usage (e.g., the memcpy and strcpy functions, from the libc library, and their sensitivity to buffer-overflow type vulnerabilities) are of interest [12]. Additionally, the use of tools providing Automated Static Analysis (ASA) alerts [10], could be useful in searching for error prone constructs during pre-release security analysis [11]. For analysis activities performed directly on software binaries, it is possible that proposed metrics from some studies [35, 40, 42], such as Source Lines of Code (SLOC) and dependency metrics (e.g., external function calls from a routine) could be modified to the viewpoint of the analyst. This would be possible by proposing the use of software binary disassembly and de-compiler tool suite features for the scenario development, in addition to compatible scripts which would gather and report the SLOC and dependency metrics[8]. It is also worth mentioning that the phase II expert judgment elicitation and Cooke classical model data analysis

[8] Binary disassemblers convert a software binary from machine code form (i.e., raw bytes of the program) into human readable assembly language for the target CPU (Central Processing Unit) and a de-compiler converts assembly language back into higher level programming languages (e.g., C or Java) [18]. It is common for these software products to include code flow graphing of various types (e.g., dependency graphs).

method could be used in a quantitative evaluation between multiple vulnerability prediction models.

Table 1-Examples Describing Variable Categories

Variable category	**Examples**
Business Decisions	Training process, resources available and their quality (personnel and tools), pre-release analysis process, information security process, rewards for responsible disclosure, design requirements (features), release frequency, and unit price
Analysis Decisions	Resources applied to analysis effort, resources available and their quality (personnel and tools), and analysis duration
External Factors	Black market rewards, peer recognition, units sold, and units retired
Design Characteristics	Quality, size, complexity, design and development artifacts
Discovery Events	Static analysis effort, dynamic analysis effort, discovery event times or counts

It is important to mention four important points related to the scope of the research. The first of which is this study will focus only on post-release security analysis estimation for consulting firms and individual, or groups of, benevolent hackers (i.e., persons whose hobbies include reverse engineering software and electronics devices). External groups also include those analyzing with malicious intent, but these will not be considered as part of this study[9]. The second item which will be considered out of scope for this study is the reward from peer recognition. Third, this research is proposing the use of the following important assumption: ASIC, initial startup firmware, and operating system software are all to be written by the same company under study. Lastly, the specific category of security faults under study is proposed to be those allowing for privileged execution of user loaded code (e.g., Android's vulnerabilities targeted by the "Killing in the Name of" or "Rage Against the Cage" public exploits [41]). In the next section is a brief discussion on metrics.

6. METRICS

This research is motivated by a primary need for software vendors to reduce the exposure of customers to software security faults in released products. Barriers to post-release detection and increased responsible disclosure are two goals related to this need. In exploring these goals, the following questions are critical: How can we discourage irresponsible disclosure[10] and how can we encourage post-release responsible disclosure? In addition to these fundamental goals, it is important to ask how we can improve the techniques for post-release vulnerability discovery modeling and how can we normalize their results when using data from various product categories and operational scenarios? This section discusses and defines several of the more important metrics relevant to exploring these questions.

One important sub-goal in the "business decisions" category is to improve the overall information security process for the software vendor. *Are critical design documents (e.g., those describing the details for CPU debugging, security, or initialization) controlled? Is access to critical design artifacts (e.g., binaries for boot-loader or security software) inhibited? Is access to critical security artifacts (e.g., private security keys for artifact encryption) controlled?* [8] Metrics for this could include public disclosure counts of sensitive design documents, design artifacts, or security artifacts.

Achieving model result consistency between different operational environments (i.e., result normalization) is a very important sub-goal related to the "discovery events" category. Post-release security analysis effort, one of the two dependent variables under study, does not have a clear definition. One approach to defining this variable is to ask the following: *What other variables is security analysis effort analogous too?* It could certainly be considered synonymous to pre-release testing effort, which in the general fault context has historically been defined as a weighted combination of man-hours plus Central Processing Unit-hours (CPU-hours) spent testing pre-release software [38]. Similarly, in the post-release user security testing context, analysis effort could be defined as the energy expended in reverse engineering software binaries and source code (i.e., static or dynamic analysis man-hours), as well as the functional testing (also known as fuzzing) of software binaries during execution (CPU-hours). The next section discusses the data analysis methods for both phases of the research.

7. DATA ANALYSIS

The Bayesian inference goals for the phase I research are parameter estimation, prediction of post-release security fault discovery events, and evaluation of various NHPP models (several intensity functions will be explored). In Bayesian estimation, model parameters are treated as the random variables. Expert judgment or relevant historical data is used to generate a prior distribution model for the parameters. Bayes theorem is then applied with current data to revise the initial assessment into a posterior distribution [16]. This posterior distribution is then used with computer simulation techniques as a source for generating information to be used in decision making.

Integration is required in the computation of the posterior distribution. When it is not possible to use a prior distribution from a conjugate family, the integration may prove very difficult. In this situation, integration estimation methods (i.e., simulations) are crucial to Bayesian analysis. Of the available simulation methods, Markov Chain Monte Carlo (MCMC) techniques are typically used in modern Bayesian inference [16]. The methods described in [22] and also [13] will be critical for the analysis in Phase I. It is hoped that the phase I results will reflect adequate performance when analyzed for predictive likelihood (i.e., measures of the "predictive power for the future given the past" [22]). However, if the phase I results are not promising, we learn that the generalized NHPP approach may not be viable for the software products and subcategories of vulnerability data under study.

[9] One justification for exclusion of malicious intent groups in this study is that in Android malware, developers have commonly used public exploits of published vulnerabilities [41], therefore they may not be expending significant security analysis effort for vulnerability discovery.

[10] Irresponsible disclosure means public disclosure without prior notification to the vendor (increases user fault exposure) [30].

The proposed analysis techniques for the phase II research are heavily influenced by a recent paper, which utilized the Cooke classical model for quantifying information security risk due to cyber-attacks [33]. Many of the same methods will be used in exploration of the related prior problem of post-release software security fault discovery[11]. Under the classical model, seed variable questions are used to calibrate the responses of experts (providing empirical control). For example, calibration questions could ask for the time to discovery of some of the more popular historical vulnerabilities enabling privilege escalation exploits, or could ask the effort required to perform a security analysis on some defined problem which has already been studied. Each expert will be scored on the calibration questions and a minimum acceptable calibration score α will be specified. Experts falling under this threshold will provide no influence on the analysis, as their data will receive zero weight. As part of the analysis, an information score (i.e., a measure of uncertainty in the answers for each expert) will also be computed for each of the experts [33].

The phase II data analysis will explore the utility of various intensity functions for NHPP models, including ones which are post release security analysis effort-dependent. Prior distributions will be developed for the fault discovery events, based on expert judgment elicited data from the different scenarios. These will be used along with computer simulation to evaluate the influences of the various control variables upon the dependent variables. Recommendations to software vendors for maximizing pre-release discovery and minimizing post-release public discovery will be based upon the data analysis results. The next section mentions several concerns to the validity of the research, with possible mitigation strategies.

8. VALIDITY THREATS, MITIGATION

Two significant threats to validity are expert judgment elicitation quality and data analysis method correctness. Interview quality assurance is an important goal for addressing the former risk. One mitigation technique will be to perform pre-testing of the expert judgment elicitation interviews using analysis experts who will not be part of the study. Another method will provide validation of the selected experts, and of the calibration questions, by performing the final data elicitation separately on a group of students [33]. Also, one problem expected with the calibration questions is that data providing the basis of calibration questions on security analysis effort estimation is more than likely scarce. Any relevant suggestions or references with data on security analysis effort are welcome.

Vulnerability data is sparse, and for that reason this research proposal includes Bayesian and expert judgment analysis techniques. Bayesian analysis will exhibit sensitivity to the chosen prior distribution; therefore, the prior distribution must be justified. One of the main purposes of the Phase I research will be to defend the prior distribution choice for the Phase II analysis. Additionally, it is intended to test the robustness of the posterior using different priors. Posterior predictive checks will be performed on the models. Lastly, for the anticipated low sample sizes, a power analysis may also be necessary [21]. In the next paragraph, the current status for the research and the upcoming schedule are briefly mentioned.

[11] Uncorrected vulnerabilities (i.e., security faults which have not been patched) are enabling mechanisms for cyber-attacks [27].

9. STATUS SUMMARY AND NEXT STEPS

The current status of the research is as follows. The data gathering and analysis technique generation for the phase I research is in progress. Analysis for the phase I research is expected to be completed in 2013. In tandem with this, the phase II interview questions are being compiled for the pre-screening study. The pre-screening study will take place on several test experts in the Fall of 2013. The anticipated time for the expert interviews is late 2013 or early 2014, with the data-analysis to follow soon after.

10. ACKNOWLEDGEMENTS

This paper describes material for portions of a dissertation to be submitted to the George Washington University in partial fulfillment of the requirements for the Doctor of Philosophy degree. The authors express sincere gratitude to all those providing input to this paper submission.

11. ADDITIONAL AUTHORS

Additional authors: Shahryar Sarkani (The George Washington University, email: emseor2003@yahoo.com), Timothy Eveleigh (The George Washington University, email: eveleigh@gwu.edu), and Thomas Holzer (The George Washington University, email: holzert@gwu.edu).

11. REFERENCES

[1] O. H. Alhazmi and Y. K. Malaiya. Quantitative vulnerability assessment of systems software. In Proceedings of the Reliability and Maintainability Symposium, pages 615-620. IEEE,Jan. 24-27, 2005. doi=10.1109/rams.2005.1408432.

[2] O. H. Alhazmi and Y. K. Malaiya. Modeling the vulnerability discovery process. In Proceedings of the International Symposium on Software Reliability Engineering, pages 10 pp.-138. IEEE,Nov. 8-11, 2005. doi=10.1109/issre.2005.30.

[3] O. H. Alhazmi and Y. K. Malaiya. Application of Vulnerability Discovery Models to Major Operating Systems. IEEE Transactions on Reliability, 57(1):14-22, Mar. 2008. doi=10.1109/tr.2008.916872.

[4] R. Anderson. Security in Open versus Closed Systems—The Dance of Boltzmann, Coase and Moore. In Proceedings of the Open Source Software : Economics, Law and Policy. IDEI,Jun. 20-21, 2002.

[5] Apple. Securities and Exchange Commission Annual Report, Form 10-K. Annual Report. Apple Inc., 2011.

[6] D. E. Bambauer and O. Day. The Hacker's Aegis. Emory Law Journal, 60(5):1051-1107, May 2011.

[7] S. Brocklehurst, B. Littlewood, T. Olovsson and E. Jonsson. On Measurement of Operational Security. IEEE Aerospace and Electronic Systems Magazine, 9(10):7-16, Oct. 1994. doi=10.1109/62.318876.

[8] C. S. Collberg and C. Thomborson. Watermarking, Tamper-Proofing, and Obfuscation Tools for Software Protection. IEEE Transactions on Software Engineering, 28(8):735-746, Aug. 2002. doi=10.1109/TSE.2002.1027797.

[9] R. M. Cooke. Experts in Uncertainty: Opinion and Subjective Probability in Science. Oxford University Press, New York, NY, 1991.

[10] M. Gegick, L. Williams, J. Osborne and M. Vouk. Prioritizing software security fortification throughcode-level metrics. In Proceedings of the Workshop on Quality of protection, pages 31-38. ACM, 1456370,Oct. 27, 2008. doi=10.1145/1456362.1456370.

[11] P. Godefroid, M. Y. Levin and D. Molnar. SAGE: Whitebox Fuzzing for Security Testing. Communications of the ACM, 55(3):40-44, Mar. 2012. doi=10.1145/2093548.2093564.

[12] R. Gopalakrishna, E. Spafford and J. Vitek. Vulnerability likelihood: A probabilistic approach to software assurance. Technical Report. Purdue University, 2006.

[13] T. Hirata, H. Okamura and T. Dohi. A Bayesian Inference Tool for NHPP-Based Software Reliability Assessment. In Proceedings of the Future Generation Information Technology, pages 225-236. Springer, Berlin, Germany,Dec. 10-12, 2009. doi=10.1007/978-3-642-10509-8_26.

[14] C. Y. Huang, S. Y. Kuo and M. R. Lyu. An Assessment of Testing-Effort Dependent Software Reliability Growth Models. IEEE Transactions on Reliability, 56(2):198-211, Jun. 2007. doi=10.1109/tr.2007.895301.

[15] IBM. IBM X-Force 2012 Mid-year Trend and Risk Report. Technical Report. IBM Security Systems, 2012.

[16] D. R. Insua, F. Ruggeri and M. P. Wiper. Bayesian Analysis of Stochastic Process Models. Wiley, 2012.

[17] H. Joh, J. Kim and Y. K. Malaiya. Vulnerability Discovery Modeling Using Weibull Distribution. In Proceedings of the International Symposium on Software Reliability Engineering, pages 299-300. IEEE,Nov. 10-14, 2008. doi=10.1109/ISSRE.2008.32.

[18] A. Johnson-Laird. Software Reverse Engineering in the Real World. University of Dayton Law Review, 19(3):843-902, Spring 1994.

[19] J. Kim, Y. K. Malaiya and I. Ray. Vulnerability Discovery in Multi-Version Software Systems. In Proceedings of the High Assurance Systems Engineering Symposium, pages 141-148. IEEE,Nov. 14-16, 2007. doi=10.1109/hase.2007.55.

[20] I. V. Krsul. Software Vulnerability Analysis. Doctoral Thesis. UMI Order Number: 9900214, Purdue University 1998.

[21] J. Kruschke. Doing Bayesian Data Analysis: A Tutorial Introduction with R. Academic Press, 2010.

[22] L. Kuo and T. Y. Yang. Bayesian Computation for Nonhomogeneous Poisson Processes in Software Reliability. Journal of the American Statistical Association, 91(434):763-773, Jun. 1996. doi=10.2307/2291671.

[23] B. Littlewood, et al. Towards Operational Measures of Computer Security. Journal of Computer Security, 2(2-3):211-229, Dec. 1993. doi=10.3233/JCS-1993-22-308.

[24] A. Mockus and D. Weiss. Interval quality: relating customer-perceived quality to process quality. ACM, City, 2008.

[25] V. H. Nguyen and F. Massacci. An Independent Validation of Vulnerability Discovery Models. In Proceedings of the Symposium on Information, Computer and Communications Security, pages 6-7. ACM, New York, NY,May 2-4, 2012. doi=10.1145/2414456.2414459.

[26] H. Okamura, M. Tokuzane and T. Dohi. Optimal Security Patch Release Timing under Non-Homogeneous Vulnerability-Discovery Processes. In Proceedings of the International Symposium on Software Reliability Engineering pages 120-128. IEEE,Nov. 16-19, 2009. doi=10.1109/issre.2009.19.

[27] H. Okamura, M. Tokuzane and T. Dohi. Quantitative Security Evaluation for Software System from Vulnerability Database. Journal of Software Engineering and Applications, 6(4):15-23, Apr. 2013. doi=10.4236/jsea.2013.64A003.

[28] A. Ozment. Software Security Growth Modeling: Examining Vulnerabilities with Reliability Growth Models. Springer, New York, NY, 2006.

[29] A. Ozment and S. E. Schechter. Milk or Wine: Does Software Security Improve with Age? In Proceedings of the Usenix Security Symposium, pages 93-104,Jul. 31-Aug. 4, 2006.

[30] A. Ozment. Improving Vulnerability Discovery Models. In Proceedings of the Workshop on Quality of Protection, pages 6-11. ACM, New York, NY,Oct. 29-Nov. 2, 2007. doi=10.1145/1314257.1314261.

[31] S. Ransbotham, S. Mitra and J. Ramsey. Are Markets for Vulnerabilities Effective? MIS Quarterly, 36(1):43-64, Mar. 2012.

[32] E. Rescorla. Is finding security holes a good idea? IEEE Security & Privacy, 3(1):14-19, Jan. 2005. doi=10.1109/msp.2005.17.

[33] J. Ryan, T. A. Mazzuchi, D. J. Ryan, J. Lopez de la Cruz and R. M. Cooke. Quantifying information security risks using expert judgment elicitation. Computers & Operations Research, 39(4):774-784, Apr. 2012. doi=10.1016/j.cor.2010.11.013.

[34] Samsung. Samsung Electronics Annual Report. Annual Report. Samsung Electronics Co., Ltd., 2011.

[35] Y. Shin and L. Williams. Can traditional fault prediction models be used for vulnerability prediction? Empirical Software Engineering1-35, Dec. 2011. doi=10.1007/s10664-011-9190-8.

[36] T. Sommestad, H. Holm and M. Ekstedt. Effort Estimates for Vulnerability Discovery Projects. In Proceedings of the Hawaii International Conference on System Science, pages 5564-5573. IEEE,Jan. 4-7, 2012. doi=10.1109/hicss.2012.238.

[37] M. J. van Eeten and J. M. Bauer. Economics of Malware: Security Decisions, Incentives and Externalities. OECD Science, Technology and Industry Working Papers, 2008/011-69, May 2008. doi=10.1787/241440230621.

[38] S. Yamada, H. Ohtera and H. Narihisa. Software Reliability Growth Models with Testing-Effort. IEEE Transactions on Reliability, 35(1):19-23, Apr. 1986. doi=10.1109/tr.1986.4335332.

[39] S. Yamada, J. Hishitani and S. Osaki. Software-Reliability Growth with a Weibull Test-Effort: a Model and Application. IEEE Transactions on Reliability, 42(1):100-106, Mar. 1993. doi=10.1109/24.210278.

[40] S. Yonghee, A. Meneely, L. Williams and J. A. Osborne. Evaluating Complexity, Code Churn, and Developer Activity Metrics as Indicators of Software Vulnerabilities. Software Engineering, IEEE Transactions on, 37(6):772-787, Nov. 2011. doi=10.1109/tse.2010.81.

[41] Y. Zhou and X. Jiang. Dissecting Android Malware: Characterization and Evolution. In Proceedings of the Symposium on Security and Privacy (SP), pages 95-109. IEEE,May 20-23, 2012. doi=10.1109/sp.2012.16.

[42] T. Zimmermann, N. Nagappan and L. Williams. Searching for a Needle in a Haystack: Predicting Security Vulnerabilities for Windows Vista. In Proceedings of the International Conference on Software Testing, Verification and Validation pages 421-428. IEEE,Apr. 7-9, 2010. doi=10.1109/icst.2010.32.